9-2012

Analyzing multi-agent systems with probabilistic model checking approach

Songzheng SONG

Jianye HAO

Yang LIU

Jun SUN
Singapore Management University, junsun@smu.edu.sg

Ho-fung LEUNG

See next page for additional authors

Citation

SONG, Songzheng; HAO, Jianye; LIU, Yang; SUN, Jun; LEUNG, Ho-fung; and DONG, Jin Song. Analyzing multi-agent systems with probabilistic model checking approach. (2012). *Proceedings of the 34th International Conference on Software Engineering: Zurich Switzerland: ICSE '12, Zurich, Switzerland, June 2-9*. 1337-1340. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/4960

Analyzing Multi-agent Systems with Probabilistic Model Checking Approach

Songzheng Song*, Jianye Hao†, Yang Liu‡, Jun Sun§, Ho-Fung Leung†, and Jin Song Dong¶
*NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore
songsongzheng@nus.edu.sg
†Department of Computer Science and Engineering, The Chinese University of Hong Kong
{jyhao, lhf}@cse.cuhk.edu.hk
‡Temasek Lab, National University of Singapore, tslliuya@nus.edu.sg
§ISTD, Singapore University of Technology and Design, sunjun@sutd.edu.sg
¶School of Computing, National University of Singapore, dongjs.comp@nus.edu.sg

***Abstract*—Multi-agent systems, which are composed of autonomous agents, have been successfully employed as a modeling paradigm in many scenarios. However, it is challenging to guarantee the correctness of their behaviors due to the complex nature of the autonomous agents, especially when they have stochastic characteristics. In this work, we propose to apply probabilistic model checking to analyze multi-agent systems. A modeling language called PMA is defined to specify such kind of systems, and LTL property and logic of knowledge combined with probabilistic requirements are supported to analyze system behaviors. Initial evaluation indicates the effectiveness of our current progress; meanwhile some challenges and possible solutions are discussed as our ongoing work.**

***Keywords*-Multi-agent systems; Probabilistic model checking**

I. Introduction

Multi-agent systems (MAS) have been successfully employed as a modeling paradigm in a number of scenarios because of their abilities in solving problems that are difficult or impossible for an individual agent to tackle. However, the existence of multiple autonomous agents in MAS makes it challenging to precisely analyze the system behaviors. Moreover, agents or the environment may have random or unreliable behaviors because of unpredictable physical conditions, which makes system analysis more difficult due to the corresponding probabilistic characteristics in MAS.

There exist mainly two approaches to quantitatively analyze the dynamics of an MAS. One is based on extensive simulation [11] while the other is based on construction of mathematical models [12]. However, the former has the drawback of inaccurate results, and the latter requires much intelligence and in some cases it is impossible to build the accurate models because of the system complexity.

In this work, we propose a novel approach to analyze MAS with probabilistic model checking. Compared with the above methods, probabilistic model checking has its unique strengths such as accurate verification results and automatic execution. We have defined an expressive modeling language called PMA (Probabilistic Multi-Agent) to specify MAS. In the current stage we assume in each round every agent only communicates with the environment instead of communicating with each other directly. Based on Markov decision processes (MDP), our approach supports a variety of properties such as reachability checking, LTL checking, reward checking and knowledge reasoning. In addition, we have implemented PMA as a part of our PAT model checking framework [8], and effectiveness of our current progress is demonstrated using one representative example.

Related work: PRISM [3] is a widely used probabilistic model checker, and some work [1] related to MAS has been done based on it. However, PRISM is not specifically designed for MAS so that some desired properties are not supported. MCMAS [5] is a symbolic model checker focusing on MAS. It supports two kinds of temporal logic and knowledge reasoning [7], and has been used in a variety of scenarios. However, it does not take probability into consideration, which limits its application in unreliable environments. Another well-known model checker is MCK [2], which is designed for analyzing the logic of knowledge in MAS. In their latest extension [4], *subjective* probability relative to agent knowledge is studied based on Discrete-time Markov Chain (DTMC). Compared with MCK, PMA focuses on a more general semantic model MDP, and supports more properties combined with probability, such as LTL checking and reward checking.

II. Background

In this section, we recall some basic concepts and definitions that are used throughout the rest of the paper.

A. MDP

MDP is popular in modeling stochastic systems which exhibit concurrency because of its ability to handle nondeterministic and probabilistic choices. Given a countable set of states S, a distribution is defined as a function $\mu : S \to [0,1]$ such that $\Sigma_{s\in S}\,\mu(s) = 1$. $Distr(S)$ is the set of all possible distributions over S. The formal definition of MDP with *action reward* extension is as follows.

Definition 1: An MDP with action reward is a tuple $\mathcal{D} = (S, init, Act, T_r, rew)$ where S is a set of states; $init \in S$ is the initial state; Act denotes the set of possible actions;

978-1-4673-1067-3/12/$31.00 © 2012 IEEE

$T_r : S \times Act \times Distr(S)$ is a transition relation, which satisfies that from one state, each enabled action has only one corresponding distribution; *rew* is a function that assigns to each action $\alpha \in Act$ a reward $rew(\alpha) \in \mathbb{N}$ where $\mathbb{N}$ is the set of natural numbers.

A state in $\mathcal{D}$ may have multiple actions enabled, which means there could be multiple distributions from one state. Therefore a schedular is used to resolve these non-deterministic choices. A DTMC can be defined as $\mathcal{D}^\delta$ given an MDP $\mathcal{D}$ and a scheduler δ; every state in $\mathcal{D}^\delta$ has just one action and distribution enabled. A rooted run in $\mathcal{D}^\delta$ is an alternating sequence of states and actions $\pi = \langle s_0, \alpha_0, s_1, \alpha_1 ... \rangle$ such that s_0 is the initial state. Suppose $(s_i, \alpha_i, \mu_i) \in T_r$, then the probability of exhibiting π in $\mathcal{D}^\delta$ is $\mu_0(s_1) * \mu_1(s_2) * \cdots$; and the *cumulative rewards* of this run is defined by $Rew(\pi) = rew(\alpha_0) + rew(\alpha_1) + \cdots$.

B. Knowledge

Reasoning about *knowledge* [7] is fundamental in MAS. Although analyzing the overall behaviors of the system is very important, in some cases such as security protocol [6] it is natural and meaningful to consider each agent's epistemic property, or its *knowledge*. An agent *knows* a fact φ if the agent could deduce φ from the information available to it.

Typically, an MAS has multiple *agents*, such as players in a game or programs in a software, and an *environment*. Assume an MAS has n agents, then its global state s has the format $(s_e, s_1, \cdots, s_n)$ in which s_e represents the state of the environment and s_i represents the agent i's local state. If an agent i has the same local state in global states s and s', then we call s and s' *indistinguishable* in the view of agent i. The definition of *knowledge* used in this paper is as follows.

Definition 2: Agent i *knows* a fact φ in global state s, which is represented by $s \models K_i\varphi$, iff for each global state s', $s' \models \varphi$ as long as s and s' are *indistinguishable* in the view of i.

III. Our Approach

In this section, we describe our approach of modeling and verifying probabilistic multi-agent systems in details.

A. Modeling Language

Modeling languages play a fundamental role in the system analysis and design. An expressive and compact language guarantees the efficiency and accuracy of modeling. PMA models have two levels: agent level, which defines the behavior of each agent and the dynamics of the environment, and system level, which regulates the composition relation between the components in agent level.

1) Agent Level: Each agent in PMA is modeled as a 3-tuple (Var, σ_i, P) based on our previous work about probabilistic model checking [10], where *Var* is a finite set of finite-domain local variables of this agent; σ_i is the initial evaluation of *Var* and *P* is a process used to define the behaviors of this agent, which supports *choices*, *interleaving* and *parallel*. We skip the language syntax because of the space limitation; interested readers could refer to [10].

```
1. #define Head 1;
2. #define Tail -1;
3. Environment{
4.  var finish = 0;
5.  behavior = if(A.coin==Head){Finished{finish=true}
6.              -> behavior} else {behavior};
7.  }
8. Agent A{
9.  var coin = 0;
9.  behavior = [Environment.finish!=true]TossCoin ->
10.             pcase{
11.                     [0.5] : {coin=Head} -> behavior
12.                     [0.5] : {coin=Tail} -> behavior
13.             }[][Environment.finish==true]Skip;
14. }
15.System = A and Environment;
```

Figure 1. Toss a coin

2) System Level: On system level we have *System* $=$ A_0 *and* A_1 *and* $\cdots$ *and* A_n *and Environment*, which indicates the participating agents and the environment. At each round, each agent A_1 to A_n could pick one action independently according to their own protocols. When they finish their actions, *Environment* could update its information according to the joint action of all agents.

A simple example is shown in Figure 1 in order to illustrate how to model with PMA. Line 1 and 2 define 2 constants *Head* and *Tail*. *Environment* is defined in line 3-7, which has a variable *finish*. Line 8-13 are the definition of an agent *A*, and *coin* is declared as a local variable in *A*, which affects the behaviors of *Environment*. Meanwhile, *A*'s behaviors are also affected by *finish*. At each round, as long as *finish* is not *true*, *A* could execute *TossCoin* to toss a coin, otherwise it will terminate. Intuitively the outcome of tossing should follow the uniform distribution, which is captured by the keyword for probabilistic choices: *pcase*. After *A*'s execution, *Environment* could choose its action according to *A*'s action choice.

B. Operational Semantics

The semantic model of PMA is an MDP because of its mixture of nondeterministic and probabilistic choices. Each global state in MDP is of the form $(s_e, s_1, \cdots, s_n)$ which is introduced in Section II-B. Because in our current setting we assume each agent is independent from others, at each round, every agent updates its local state according to its behavior's operational semantics, which is also defined in [10]. After every agent updates its local state, *Environment* will execute an action accordingly and transfer to another global state. Suppose at the agent level we have $s_i \xrightarrow{a_i} \mu_i \wedge \mu_i(s_i') > 0$, then for the global state we have $(s_e, s_1, \cdots, s_n) \xrightarrow{a} \mu$, where $a = (a_1, \cdots, a_n, a_e)$ is the joint action of each agent and the

environment, meanwhile $\mu((s'_e, s'_1, \cdots, s'_n)) = \mu_1(s'_1) \times \cdots \times \mu_n(s'_n) \times \mu_e(s'_e)$.

C. Verification

Our approach supports multiple kinds of properties, each of which focuses on a specific aspect of MAS. On one hand, we use reachability checking, LTL checking and reward checking to analyze the overall behavior of the system. On the other hand logic of knowledge is supported to check agents' epistemic properties. For property specification, $P_r(system \models \varphi)$ is used to specify the probability that a property φ is satisfied by the system, where φ could be propositions, LTL formulas and epistemic formulas; another property is $R(system \models \gamma)$ calculating the average cumulative rewards of reaching some target states. Notice that typically an MDP has many, or even infinite, schedulers and corresponding DTMCs, so here we calculate the **maximum/minimum** probability/rewards.

Due to the space limitation, we focus on the knowledge reasoning algorithm used in PMA. Currently we have finished a simple combination of linear temporal operator and $K_i\varphi$ where i is the index of an agent and φ is a proposition. For example $P_r(system \models \Diamond K_i\varphi)$ represents the probability of reaching a state where agent i *knows* φ from the initial state of the system. In order to apply the probabilistic model checking, given a PMA system and a property, we should build the corresponding MDP and at the same time find out the **target** states. This progress is depicted in Algorithm 1. Starting from the initial state of PMA model, we will generate the whole MDP step by step according to our operational semantics. Note that we will group the states having the same local state of agent i together. *Ts* and *NTs* are defined as two sets of these group states. For $\forall T \in Ts \Rightarrow (\forall t, r \in T, t_i = r_i) \ \wedge \ (\forall s \in T, s \models \varphi)$. For $\forall NT \in NTs \Rightarrow (\forall t, r \in NT, t_i = r_i) \ \wedge \ (\exists s \in NT, s \not\models \varphi)$. Another point we want to emphasize is *act* enabled in s. Here *act* is a joint action instead of an individual action in $\mathcal{M}$.

After building the MDP and finding out the target states, this knowledge reasoning problem is converted to probabilistic reachability checking, which could be solved as a linear program.

Algorithm 1: BuildMDP for $P_r(system \models \Diamond K_i\varphi)$

Input: PMA $\mathcal{M}$ and property $P_r(\mathcal{M} \models \Diamond K_i\varphi)$;
Output: MDP $\mathcal{D}$ with target states;
s_{init} := initial state of $\mathcal{M}$;
let *Ts*, *NTs*, *Act*, $T_r := \phi$;
let *visited*, *working* := $\{s_{init}\}$;
while *working* $\neq \phi$ **do**
  let $s \in$ *working*;
  working := *working* $\setminus \{s\}$;
  forall the *action act enabled in s* **do**
    let $\mu := s \rightarrow act$;
    add *act* to *Act* and (s, act, μ) to T_r;
    forall the $\mu(r) > 0$ *and* $r \notin$ *visited* **do**
      add r to *visited* and *working*;
      if $r_i = nt_i$ *for* $nt \in NT$ *and* $NT \in NTs$ **then**
        add r to *NT*;
      else if $r_i = t_i$ *for* $t \in T$ *and* $T \in Ts$ **then**
        if $r \models \varphi$ **then**
          add r to T;
        else
          $Ts = Ts \setminus \{T\}$;
          add r to T;
          add T to *NTs*;
      else if $r \models \varphi$ **then**
        add $\{r\}$ to *Ts*;
      else
        add $\{r\}$ to *NTs*;
States in elements of *Ts* are labeled as **target** states;
Result: $\mathcal{D} := (visited, s_{init}, Act, T_r)$;
//For simplicity, *rew* is ignored.

IV. Preliminary Evaluation

We have implemented PMA into our model checking framework PAT. In this section, we show the effectiveness of our approach using the dispersion game [11] from game theory domain. Dispersion game is the generalization of anti-coordination game to an arbitrary number of players and actions, which has received wide attention in many applications. We focus on one novel strategy designed for dispersion games: extended simple strategy (ESS) in this evaluation. In each round, every player in the system will pick one action according to specific probabilistic distribution. In dispersion game, the desired outcome is called Maximal Dispersion Outcome (MDO), and we are interested in analyzing properties related to MDO under ESS strategy. Techniques in [9] are used to reduce the state space of this model and in our experiments we assume initially all agents choose the same action. Note that the semantic model of this example is a DTMC, but PMA could support MDP.

A. Convergence to MDO

We first consider whether the agents adopting ESS are guaranteed to converge to an MDO. We express this property using $P_r(System \models \Diamond\Box MDO)$, which means finally the *system* will stay in an MDO forever. The results are shown in Table I, where n is the number of players and k means the number of actions. We could see that in some cases, the outcome will converge to MDO while in others it cannot.

B. Departure from MDO

Because in some cases ESS cannot converge to an MDO, it is interesting to check the probability that the system

Table I
PROBABILITY OF CONVERGENCE TO AN MDO OF ESS

System	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
k=2	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0
k=3	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
k=4	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0

Table II
PROBABILITY OF DEPARTURE AFTER REACHING AN MDO

System	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
k=2	0.063	0.0	0.070	0.0	0.075	0.0	0.069	0.0
k=3	0.0	0.072	0.12	0.0	0.091	0.14	0.0	0.092
k=4	0.0	0.0	0.12	0.15	0.16	0.0	0.14	0.16

deviates from MDO after reaching it. We use a reachability checking to analyze this property, which is expressed as $P_r(system \models \Diamond Depart)$. Here *Depart* is a proposition which represents departure from MDO. The verification results are displayed in Table II, which are consistent with the results in Table I since convergence probability 1 indicates the probability of departing from MDO is 0.

C. Average Rounds to MDO

Another interesting property is that how many rounds does the ESS system take to reach an MDO, which could be verified using reward checking $R(system \models \Diamond Depart)$. Intuitively, if we set the action in the *Environment* of ESS having reward 1, then after each round, the *cumulative rewards* is increased by 1. Using the iterative method we mentioned in Section III-C, we could get the average rounds, or rewards, from initial state to MDO. The results are shown in Table III.

Table III
AVERAGE ROUNDS TO MDO

System	n=3	n=4	n=5	n=6	n=7	n=8	n=9	n=10
k=2	1.33	2.44	1.55	2.69	1.70	2.87	1.81	3.00
k=3	2.63	1.48	2.11	3.20	1.81	2.45	3.52	2.04
k=4	2.15	3.08	1.58	2.15	2.90	3.73	2.04	2.59

V. RESEARCH CHALLENGES

The development of PMA reveals two research challenges. First, our approach requires that each agent should communicate with the *Environment* instead of communicating with each other directly. This requirement could cover a lot of cases, but there also exist many cases in which agents affect each other directly. These scenarios will increase the complexity of system analysis since different orders of the actions between agents will generate different global states. We are now trying to define more suitable operational semantics for tackling the actions interleaving in MAS.

Second, till now we just support $K_i\varphi$, where φ is proposition. There are other kinds of knowledge such as $E_G\varphi$ (everyone in group G knows φ) and $C_G\varphi$ (φ is common knowledge in G). Combining these knowledge operators with temporal logic and probability is quite challenging. Besides, *subjective* probability in knowledge reasoning such as $K_i(P_r(\varphi) > b)$, where b is a probability threshold, also deserves our exploration.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel approach to analyze MAS. Compared to the existing approaches, ours supports a more expressive language, with which we could efficiently build accurate and compact models which have stochastic characteristics; quantitative calculations for different kinds of properties guarantee that many aspects of the system could be accurately analyzed. Preliminary evaluation demonstrates the ability of PMA in modeling and verification.

In the future, we will focus on two issues that we mentioned in Section V: (1) investigating multi-agent systems whose agents have dependency between each other, and (2) further exploring the combination of probability and logic of knowledge, which are useful in different multi-agent interaction scenarios.

REFERENCES

[1] T. Chen, M. Kwiatkowska, D. Parker, and A. Simaitis. Verifying team formation protocols with probabilistic model checking. In *CLIMA XII*, pages 190–297, 2011.

[2] P. Gammie and Ron van der Meyden. Mck: Model checking the logic of knowledge. In *CAV*, pages 479–483, 2004.

[3] A. Hinton, M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. In *TACAS*, pages 441–444, 2006.

[4] X. Huang, C. Luo, and R. Meyden. Symbolic model checking of probabilistic knowledge. In *TARK*, pages 177–186, 2011.

[5] A. Lomuscio and F. Raimondi. Mcmas: A model checker for multi-agent systems. In *TACAS*, pages 450–454, 2006.

[6] R. Meyden and Kaile Su. Symbolic model checking the knowledge of the dining cryptographers. In *CSFW*, pages 280–291, 2004.

[7] Y. Moses R. Fagin, J. Y. Halpern and M. Y. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[8] J. Sun, Y. Liu, J. S. Dong, and J. Pang. PAT: Towards Flexible Verification under Fairness. In *CAV*, pages 709–714, 2009.

[9] J. Sun, Y. Liu, A. Roychoudhury, S Liu, and J. S. Dong. Fair model checking with process counter abstraction. In *FM*, pages 123–139, 2009.

[10] J. Sun, S. Z. Song, and Y. Liu. Model Checking Hierarchical Probabilistic Systems. In *ICFEM*, pages 388–403, 2010.

[11] Y. Shoham T. Grenager, R. Powers. Dispersion games: general definitions and some specific learning results. In *AAAI*, pages 398–403, 2002.

[12] J. M. Vidal and E. H. Durfee. Predicting the expected behavior of agents that learn about agents: The clri framework. *AAMAS*, 6:77–107, 2003.